Optimax Energy – trading bot test application.

Project description.

Trading bot application has been written as a part of code challenge task. The main goal of this application is developing one or more implementation of the given Bidder interface that could provide algorithm(s) to win in auction.

Application's source code could be found on GitHub by the following URL:

https://github.com/viktar-lebedzeu/bidders

Project has standard Apache Maven structure. Build, Unit testing and generation of some additional reports are also organizing by Maven.

Run '`mvn clean install`' to build the project and run unit tests.

Some additional reports can be found in target/site folder:

- apidocs contains generated JavaDoc for project classes
- jacoco contains code coverage report generated by JaCoCo

Application is written on Java 8 using Spring Boot project and uses few additional 3rd-party libraries:

- Lombok for simplifying declaration of class constructors, getter and setter methods, logger instances
- Apache Commons Lang to simplify working with standard java classes, e.g. String
- Apache Commons CLI for better processing and support of command line options

Running application.

To run the application use the following command:

```
java -jar bidder.jar -b1 <BIDDER 1> -b2 <BIDDER 2> -c <MONETARY UNITS (MU)> [-h] -qty <QUANTITY UNITS (QU)> [-v]
```

Application options:

-b1,	--bidder1 <BIDDER 1>	Type of the first bidder [equal, weighted, weighted-correction]
-b2,	--bidder2 <BIDDER 2>	Type of the second bidder [equal, weighted, weighted-correction]
-c,	--cash <MONETARY UNITS (MU)>	Initial amount of money
-h,	--help	Prints help message
-qty,	--quantity <QUANTITY UNITS (QU)>	Quantity of product items to be auctioned
-v,	--verbose	Turn on verbose messages

Description of bidder types (strategies).

Current project provides three implementations of bidder strategies. They are described below.

• "equal" type

The simplest algorithm. Based on splitting whole amount of money on equal parts.

For example if we have 100 MU and 20 QU initially then every bid (auction lot) will have value = 10 MU.

Obviously, it is the weakest algorithm and implemented for testing purpose only.

• "weighted" type

More efficient algorithm.

According to auction rules, bidder who will got more QU then other one wins. It means that bidder should achieve 50% + 1 QU of the initial QU amount. Every lot that makes the bidder closer to final goal must be more valuable for him.

Using previous example the final goal is receiving (20 * 50% + 1) = 11 QU.

Beginning from the first bid = 10MU the value of every next auction turn will grow exponentially to have bigger win probability.

• "weighted-correction" type

A little bit improved algorithm based on the previous one. Counts percentage of tie turns. When percentage reaches threshold (e.g. 50%) bidder adds some additional value to calculated bid to increase chance of win.